MA2-IRIF

[INFO-H413] - Heuristic Optimisation

Implementation Exercice 2

https://github.com/Mudezer/LOP-VND

Author
Loïc Bermudez Arroyo

Professor

Thomas Stützle

Christian Camacho-Villalón

2023-2024

Abstract

This report focuses on the implementation exercise which is an attempt of solving the Linear Ordering Problem through different algorithms and comparing their efficiency and solution quality.

This work is divided in two parts, the first one focusing on studying two stochastic local search algorithms, the first one being a hybrid SLS algorithm and the second one being a perturbative population-based algorithm. The study was about the choice of their components and the best values for their respective parameters. The second part focused on comparing them through the Wilcoxon paired test in order to highlight whether the two compared algorithms are different or not statistically, but also through a correlation plot to study their correlation.

The results have shown that the hybrid SLS algorithm had a upper hand in term of solution quality on the genetic population-based one, their correlation heavily depends on the instance characteristics they are run onto.

Contents

1 Introduction

1.1 Introduction

The Linear Ordering Problem has a wide range of application in several fields, such as ranking in sport tournaments, scheduling, archeology, and economics making it an active combinatorial research subject. Due to its combinatorial nature, it has been shown to be NP-Hard, thereby there is no precise algorithm for solving LOP instances efficiently. As of today, researchers continue to work on it, thus collecting a significant amount of experiment and data making easy to compare algorithms.

Linear Ordering Problem

In this work, the Linear Ordering Problem we are addressing is conceptually simple:

Given a nxn matrix C, where the value of row i and column j is noted c_{ij}. Find a permutation π of the column and row indices {1,...,n} such that the value f(π) = $\sum_{i=1}^{n}\sum_{j=i+1}^{n} c_{\pi(i)\pi(j)}$ is maximized.

In other words, the goal is to find a permutation of the columns and rows of C such that the sum of the elements in the upper triangle is maximized.

1.2 Specifications

In this experiment on Linear Ordering Problems we used the XLOLIB instance class which is a database composed of 78 instances of two possible sizes: n=150 and n=250. We focused resolving the instances of size 150 with two kinds of stochastic local search, i.g. hybrid iterative local search algorithm and population based local search algorithm. In order to compare the different algorithms, the Wilcoxon Paired Test has been used by using the relative deviation between the results and the best knowns of the instances.

1.3 Objectives

This work has been split in two parts:

1. Implementing two different type of stochastic local search algorithms and study their different parameters and components in order to optimize them.

2. Comparing the two different SLS algorithms by computing their relative deviation from the best known of each instance, and showing through a statistical test, if there is a statistical difference between the solutions generated by both algorithms. And finally presenting correlations plots between both algorithms.

1.4 Methods

For the instance solving two stochastic local search algorithms were implemented:

- An hybrid SLS algorithm: Iterative Local Search
- A population based algorithm: Genetic Local Search (or Memetic Algorithm)

For the statistical tests, we opted of the Wilcoxon signed-rank test (Wilcoxon paired test) which allows to show whether there is a statistical difference between the results of two different algorithms. A statistical test is based on a statement to be tested, in the Wilcoxon paired test this statement also said the null hypothesis is that the median of the differences between two sets is zero".

For the test a significance level α determining the maximum allowable probability of incorrectly rejecting the null hypothesis is chosen. Here we've chosen a α of 0.05 (typically the value are 0.05 or 0.01). This α will be compared to the p-value which the test returns, the p-value represents the probability that the null hypothesis is incorrectly rejected. If the p-value is smaller than the significance level, it means that the probability of falsely rejecting the null hypothesis is accepted.

In other words the smaller the p-value the more it shows that there is a difference between both sets. As the significance level plays a role of a threshold, if the p-value it this report is under 0.05 it then means that there is a statistical significant difference between the values we are observing.

2 Algorithms

This section focus on the description and the study of the components and parameters for the two implemented algorithms. The goal is to explain how they work, what components have been chosen and which parameters have been selected to conduct the comparison.

Termination Criterion

SLS algorithms require a termination criterion, as the stochastic behavior allows to explore the search space there is a risk to run for a very long time or worse, to never stop running. A termination criterion needs to be sufficiently flexible to allow the algorithm to efficiently explore the search space but also strict to avoid being near never ending.

Therefore a maximum run time criterion was implemented as asked, the running time being instance dependent computed from the running time of Piped Variable Neighbour Descent algorithm implemented in the first implementation exercise. The maximum run time for an instance is then the mean of the run time for both VND sequence multiplied by 100. See **Table 2.1** for the different run times.

2.1 Iterated Local Search

As first algorithm an Iterated Local Search was implemented, from the "Hybrid" SLS class, it tries to escape local optima in order to improve the solution quality through combination of two types of SLS steps. It then generally finds a better local optima or in best case, the global optima.

Instance	MaxTime [s]
N-be75eec_150	22.0727
N-be75np_150	22.84685
N-be75oi_150	31.82815
N-be75tot_150	22.3964
N-stabu1_150	23.7299
N-stabu2_150	24.57105
N-stabu3_150	23.97835
N-t59b11xx_150	23.187
N-t59d11xx_150	24.21895
N-t59f11xx_150	19.57155
N-t59n11xx_150	24.7623
N-t65b11xx_150	26.25305
N-t65d11xx_150	18.8593
N-t65f11xx_150	22.0227
N-t65l11xx_150	21.7412
N-t65n11xx_150	21.9899
N-t69r11xx_150	27.57895
N-t70b11xx_150	26.32435
N-t70d11xn_150	21.7334
N-t70d11xx_150	23.1466
N-t70f11xx_150	17.6518
N-t70l11xx_150	32.1813
N-t70n11xx_150	24.37505
N-t74d11xx_150	20.8891
N-t75d11xx_150	24.3235
N-t75e11xx_150	22.89345
N-t75k11xx_150	27.1095
N-t75n11xx_150	23.60715
N-tiw56n54_150	21.1215
N-tiw56n58_150	22.8186
N-tiw56n62_150	22.41225
N-tiw56n66_150	20.65155
N-tiw56n67_150	23.69545
N-tiw56n72_150	23.76775
N-tiw56r54_150	20.4552
N-tiw56r58_150	24.2724
N-tiw56r66_150	20.04465
N-tiw56r67_150	28.8173
N-tiw56r72_150	22.41135

Table 2.1: Maximum run time in seconds per instance

2.1.1 Description

The Iterated Local Search is based on two types of SLS steps:

- Intensification: make use of subsidiary local search steps in order to reach a local optima as efficiently as possible
- Diversification: make use of perturbation steps in order to effectively escaping from the local optima (also known as perturbation step)

The ILS will then alternate between intensification and diversification through the whole execution in order to find the best solution possible:

After initialization the candidate solution will be intensified to improve it. The algorithm will then diversify the current solution to escape local optima, followed by another intensification to refine the perturbed solution. The process repeats until a termination criterion is met, and at each iteration, the algorithm decides whether to accept the new solution or keep the current one based on an acceptance criteria.

Diversification

The diversification is the most important element of the iterated local search. It is directly responsible for escaping the current local optima. Each diversification step is a succession of simple perturbation steps, the number of steps (strength of the perturbations) is to be wisely chosen. If it is too high (too strong), the diversification behave like a random restart, thus not allowing any improvement. It it is too low, the local search may undo the perturbation easily, thus looping in the same neighborhood and always returning to the same local optima.

2.1.2 Components

Initialization

As a perturbation based algorithm, the Iterative Local Search requires an input candidate to perform perturbations on. The initialization consist solely in generating a starting point to the algorithm.

Local Search

In the intensification phase of the Iterated Local Search, the goal is to find the best candidate solution possible after the initialization and after each perturbation step, with the expectation that it will lead to a better candidate on the next iteration.

Perturbation

The Diversification is the sole source of perturbation in the Iterated Local Search, it aims at find new candidate solution by escaping local optima from the current solution. In this implementation, the perturbation operation is applied multiple times on randomly chosen component of the current candidate solution.

Acceptance Criterion

In order to accept a new candidate or to keep the current solution a greedy criterion as been implemented. In other words, a new candidate will be kept for the next iteration only and only if has a better cost than the current one.

AcceptanceCriterion(new,current*) : accept new* only if* $f(new*) > f(current*)$ with f, the objective cost function we search to maximize.

2.1.3 Parameters

Initialization

For the initialization of the algorithm, a choice has to be made between a simple Chenery and Watanabe greedy heuristic or a random initialization. As a perturbative hybrid local search algorithm looking to escape from local optima by using his perturbation, it would make better use of a greedy initial solution which would give a better starting point for the exploration of the search space. On another hand, a randomly generated initial candidate would take advantage of the stochastic perturbations aspect of the algorithm, thus a preliminary study has been made.

Initialization	Mean Deviation [%]
CW	0.868456002405613
Random	0.895691394771626

Table 2.2: Mean deviation from best-known solution w.r.t. Initialization method

From the study we can see that, the greedy heuristic initialization gives the best result, this initialization will thus be chosen.

Local Search

As stated previously, the goal of the Local Search is to find the best local optima attainable with the current candidate, therefore we have opted for the Piped Variable Neighbour Descent. Based on the previous implementation exercise, it has been proven that the VND gives better solution quality in comparison with simple Iterative Improvement algorithms like the best and first improvement. It typically gives much better solution quality thanks to the multiple neighbours searches. The sequence chosen for the pipeline is the **transpose, exchange, insert** as it typically gives much better results than the **transpose, insert, exchange**. See the **Table 2.3** coming from the first implementation exercise.

Algorithm	Size	Mean Deviation [%]
vnd_first_cw_TEI	150	2.65762453832907
vnd_first_cw_TIE	150	3.82220234217974

Table 2.3: Mean deviation from best-known solution w.r.t. Neighbourhood sequence

Perturbation

The perturbation operations should ideally be complementary to the Local Search and not too simple for the Local Search to undo. The operation choice as been made between two different options[1]:

- Exchange: randomly chose two candidate solution components to exchange.
- Insert: randomly chose one candidate solution component to insert to a randomly chosen place.

A preliminary experiment has been conducted by running the ILS on each of the three possibilities, in order to decide which one to take.

Operation	Mean Deviation [%]
Exchange	1.22936716318559
Insert	0.875556167165916

Table 2.4: Mean deviation from best known solution w.r.t. Perturbation Operation

This experiment clearly shows the superiority of the insert operation.

Number of Moves

Each diversification phase is a succession of multiple simple perturbation steps, the number of simple steps define the strength of the perturbations. As previously stated, if too strong the diversification is close to a random restart thus undoing all previous improvement and if too weak the local search may undo the perturbations too easily whatever the operation. To determine the optimal number of steps (i.e. moves), the algorithm was run on each instance with different values of moves (2,3,4,5,6,7,8,9). The results shown in the **Table 2.5** clearly state that the best number of moves is of 4.

[1]The transpose operation was not considered, as it is too easy to undo and is an exchange operation with additional constraints.

Moves	Mean Deviation [%]
2	0.87525044603254
3	0.906931360717833
4	0.869307810653952
5	0.998275757251968
6	0.877878494561741
7	0.920164610085568
8	0.962851780199739
9	1.00144855608486

Table 2.5: Mean deviation from best known solution w.r.t. Number of Simple Steps

2.2 Memetic Algorithm

As second algorithm a Memetic Algorithm was implemented, from the Population-based SLS class, it uses a "population" of candidate solutions instead of a single one at a time like the Iterated Local Search. This alone is a big advantage for better search space exploration.

2.2.1 Description

A Genetic Local Search Algorithm is built on applying iteratively genetic operators such as mutation, recombination and selection to mimic the principles taking place in evolution. The difference from the Evolutionary Algorithm is that it applies a subsidiary local search after each population modification allowing better search intensification.

2.2.2 Components

Initialization

Being a perturbative population-based algorithm, the Memetic Algorithm requires an input population to perform perturbations on. The initialization step consist simply in generating a starting point to the algorithm.

Subsidiary Local Search

The subsidiary local search is the component in which the Memetic Algorithm differs from the Genetic one as it gives an intensification behaviour to it. The aim is to improve the quality of the population before each evolutionary step, with the expectation that it will lead to a better population on the next iteration.

Recombination

The main source of perturbation in the Memetic Algorithm is the Recombination process, the goal is to generate new individuals based on the actual population. In a population-based algorithm the aim is to have as much diversity in the population, the more diversity the more powerful is the perturbation coming from the recombination. In this implementation, the recombination mechanism is a two-point crossover applied randomly to one of the best individuals (top 25%) and an another randomly chosen individual in the population. This elitist aspect is due to the desire to generate the best possible candidate population, we have more chance to generate better candidate if the recombination is based on at least one of the best individuals from the current population than on two randomly chosen individuals.

Mutation

The Mutation is the second source of perturbation in the algorithm, it aims to avoid stagnation of the algorithm in a local optimum. In fact, even if the recombination increases diversity, individuals remain relatively similar due to the sharing of candidate components from the crossover which aims to keep the best individuals. Therefore, the mutation is an extremely important operation to increase the diversity in a relative similar population.

In this implementation, a random exchange operation is performed on each individual with a probability equal to the mutation rate.

Selection

This process emulates the natural selection observed in evolutionary algorithms. Its objective is to retain the most proficient individuals from each generation to seed the subsequent one. Only those selected get the opportunity to contribute to the next generation through recombination, ensuring that only the finest solutions perpetuate their inherent traits forward.

In this implementation, we preserve the top N individuals, drawn from both the current and preceding generations, to constitute the population for the next iteration. Here, N represents the population size.

2.2.3 Parameters

Initialization

In this case, the use of a deterministic greedy constructive heuristic will cause zero genetic diversity, going against the purpose of the population idea. For this reason, the initialization is an association of N solution candidates randomly generated. Here, N represents the population size.

Subsidiary Local Search

In order to limit the execution time and optimize the solution quality, an Iterative Improvement SLS algorithm has been used with first improvement. The neighbourhood was chosen by running, on all the instance, the Memetic algorithm on three options and choosing the neighbourhood returning the best mean deviation.

Neighbourhood	Mean Deviation
Exchange	1.5150905725353
Insert	1.63423423681914
Transpose	3.19587955641965

Table 2.6: Mean deviation from the best known w.r.t. Neighbourhood

This study clearly shows that the exchange neighbourhood gives the best solution quality.

Population Size

Because of the initial randomization, the population size serves as first main driver of genetic diversity. A larger population provides a richer pool of genetic material for generating new individuals. However, it also entails increased computational overhead for genetic operators. Thus a preliminary experiment was conducted to select the best population size on all the instances.

Population Size	Mean Deviation
10	1.41287139448583
20	1.32523316947433
50	1.52949705199588
100	1.80262598040991
200	1.98411807563732

Table 2.7: Mean deviation from the best known w.r.t. Population Size

From this experiment we can see that, the best population size is 20 individuals, this parameter will thus be chosen such that the computation.

Mutation Rate

The mutation rate is the second source of genetic diversification. To determine the optimal mutation rate, 11 experiments were launched for different values of mutation rate (0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0).

Mutation Rate [%]	Mean Deviation [%]
0	1.69905182593983
0.1	1.47195133034776
0.2	1.37095962330211
0.3	1.33141882818365
0.4	1.44828328741707
0.5	1.45041276112412
0.7	1.35705526668565
0.9	1.36431342483987
1	1.40719977176914

Table 2.8: Mean deviation from the best known w.r.t. Mutation Rate

The experiment results displayed in the **Table 2.8**, show that the best mutation rate is at 0.3

Remark on the Max Run Time Criterion

The addition of a subsidiary local search mixed with a population of candidates creates an explosion of large time overheads to optimize each individuals after each genetic operator. The biggest time consumption is right after the random initialization where the longest optimization takes place as each candidates need to be optimized from random configurations. Therefore for the algorithm to start working on the genetic processes we need to give it enough time after initialization. We then started to count the maximum run time right after the first subsidiary local search and the initialization, allowing the algorithm to fully exploit the genetic behaviour and give supposely better results.

3 Comparison

3.1 Results

The results can be found in the **Table 3.2**.

3.2 Statistical Tests

Mean Deviation ILS [%]	Mean Deviation Memetic[%]
0.870115781277832	1.34726489446107

Table 3.1: Mean Deviation from best-known solution w.r.t. the Algorithm

The p-value returned by the paired Wilcoxon test is : 3.288732e-09. Which corroborates the difference between the results of both algorithms. From a statistical point of view, the Iterated Local Search Algorithm is better than the Memetic Algorithm.

3.3 Correlation Plot

Hereafter come the correlation plot made with the results of the previous comparison to the best-known solutions. To the correlation plot is added (in black) a line which is the linear regression using the least-squares of the dots. This line is supposed to be the best possible linear fit for the given data.

In this correlation there is a majority of dots being quite near the line and even one that seem to be perfectly on the linear regression line, this can indicate a potentially strong correlation . While others are really far from the trend line, indicating that the relation ship might not hold as strongly, possibly due to specific characteristics of the instances. Furthermore as the ILS deviation increases so does the Memetic deviation, indicating a positive correlation.

In fine, the correlation is not clearly visible and one could argue that the distribution is linear. The correlation seem to heavily depend on the instance characteristics.

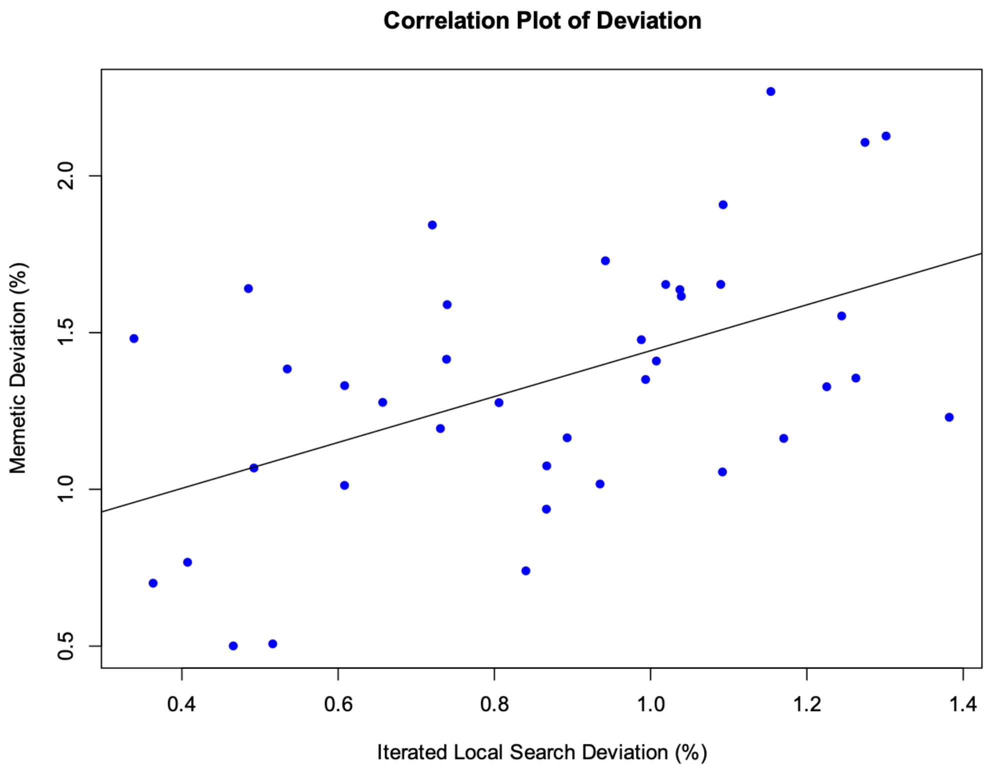

Figure 3.1: Correlation plot of the Deviation from best-known solution

Instance	Deviation ILS [%]	Deviation Memetic [%]
N-be75eec_150	0.866652042535549	0.936824902062347
N-be75np_150	0.608182651329568	1.01270851264753
N-be75oi_150	0.363137170414514	0.700768383652328
N-be75tot_150	1.27451769479855	2.10664202849238
N-stabu1_150	0.867083580806556	1.07503063567815
N-stabu2_150	0.516344397206911	0.507216805490789
N-stabu3_150	0.534891790056192	1.3839654402171
N-t59b11xx_150	0.338627278480036	1.48091555925977
N-t59d11xx_150	1.22547595298274	1.32747930706086
N-t59f11xx_150	1.00742809799363	1.40944062218013
N-t59n11xx_150	1.09198645598194	1.05561825934286
N-t65b11xx_150	0.739699614306902	1.58917890073438
N-t65d11xx_150	1.26284399919423	1.35479906848082
N-t65f11xx_150	1.24469586234532	1.55311607602381
N-t65l11xx_150	0.407267675890701	0.767573284503307
N-t65n11xx_150	0.73883351823842	1.41502325083117
N-t69r11xx_150	0.805923975601463	1.27665780164351
N-t70b11xx_150	0.657017678340589	1.27741344172508
N-t70d11xn_150	1.30146071850204	2.12686478126646
N-t70d11xx_150	1.09287422552443	1.90783291314245
N-t70f11xx_150	0.72070487216066	1.84320800171337
N-t70l11xx_150	0.840311127999231	0.740279539076413
N-t70n11xx_150	1.38242388922904	1.22993361878275
N-t74d11xx_150	0.942226324184944	1.72901489179914
N-t75d11xx_150	1.01946248446922	1.65351260197425
N-t75e11xx_150	0.935343263862162	1.01715909762555
N-t75k11xx_150	0.465815946590417	0.500779711415961
N-t75n11xx_150	1.03935874944208	1.61586236886823
N-tiw56n54_150	0.485234711108724	1.64056113468068
N-tiw56n58_150	0.988149476541295	1.47724754888384
N-tiw56n62_150	0.73101275353874	1.19397315542672
N-tiw56n66_150	1.1705151642683	1.16263897791774
N-tiw56n67_150	0.993709875487057	1.35048459159704
N-tiw56n72_150	1.0376200431973	1.63729503588935
N-tiw56r54_150	0.608333194182377	1.33094411140982
N-tiw56r58_150	0.492251670022431	1.06815823898236
N-tiw56r66_150	1.08983359568828	1.65378937578417
N-tiw56r67_150	0.893234458970203	1.16449217957864
N-tiw56r72_150	1.15402948836267	2.2689267281403

Table 3.2: Deviation from best-known solution w.r.t. the Instance

4 Conclusion

This work started by introducing the Linear Ordering Problem and the two Stochastic Local Search methods by explaining their different components and parameters. Then followed the identification of the best values for these parameters. Finally the two algorithms were run configured with the previously found parameters values to present their relative efficiency by comparing their output with the best-known solution.

To conclude, the Iterative Local Search is statistically better than the Memetic Algorithm on the given instances and with the given configurations. Also it was shown that the correlation between the results of those algorithms is heavily instance dependent, while on a lot of instances there is a strong correlation, on some instances the correlation seem to be quite weak. This surely come from the fact that instances are not similar, and the inherent characteristics quite affect the performance of both algorithms.